Peer-to-Peer Networks

Chapter 2: Initial (real world) systems

Thorsten Strufe

Chapter Outline

- Overview of (previously) deployed P2P systems in 3 areas
- P2P file sharing and content distribution:
  - BitTorrent, Napster, Gnutella, KaZaA
  - Differences, strengths, weaknesses
- P2P Communication
  - Typical instant messaging setup
  - Skype
- P2P Computation
  - SETI@Home example

Current P2P Content Distribution Systems

- Most intial P2P content distribution systems targeted at one application: File sharing
- Users share files and others can download them
- Content typically music, videos, or software
  - Also often illegally shared… :-(
  - Legal uses becoming more common? (see BitTorrent)
- Content distribution has made P2P popular

- Note: Distinguish between name of network (e.g., BitTorrent) and name of client (e.g., Vuze)

BitTorrent

- BitTorrent is an approach to sharing large files
- BitTorrent used widely also for legal content
  - For example, Linux distributions, software patches
  - Official movie distributions (WB)
  - BBC series over the Internet (iPlayer), Octoshape
  - Game distribution (Pando Networks, Akamai's Netsession Interface)
- Goal of BitTorrent:
  - Quickly and reliably replicate one file to a large number of clients
- BitTorrent more appropriately called "peer-to-peer content distribution"

P2P Content Distribution

- BitTorrent builds a network (swarms) for every file that is being distributed

Big advantage of BitTorrent:

- Can send "link" (.torrent) to a friend
- "Link" always refers to the same file (remember identifiers?)
- Same not really feasible on Napster, Gnutella, or KaZaA
  - These networks are based on searching, hard to identify a particular file
  - Downside of BitTorrent: No searching possible
    - Websites with "link collections" and search capabilities exist
    - → BitTorrent implements location- but no name service…

BitTorrent History

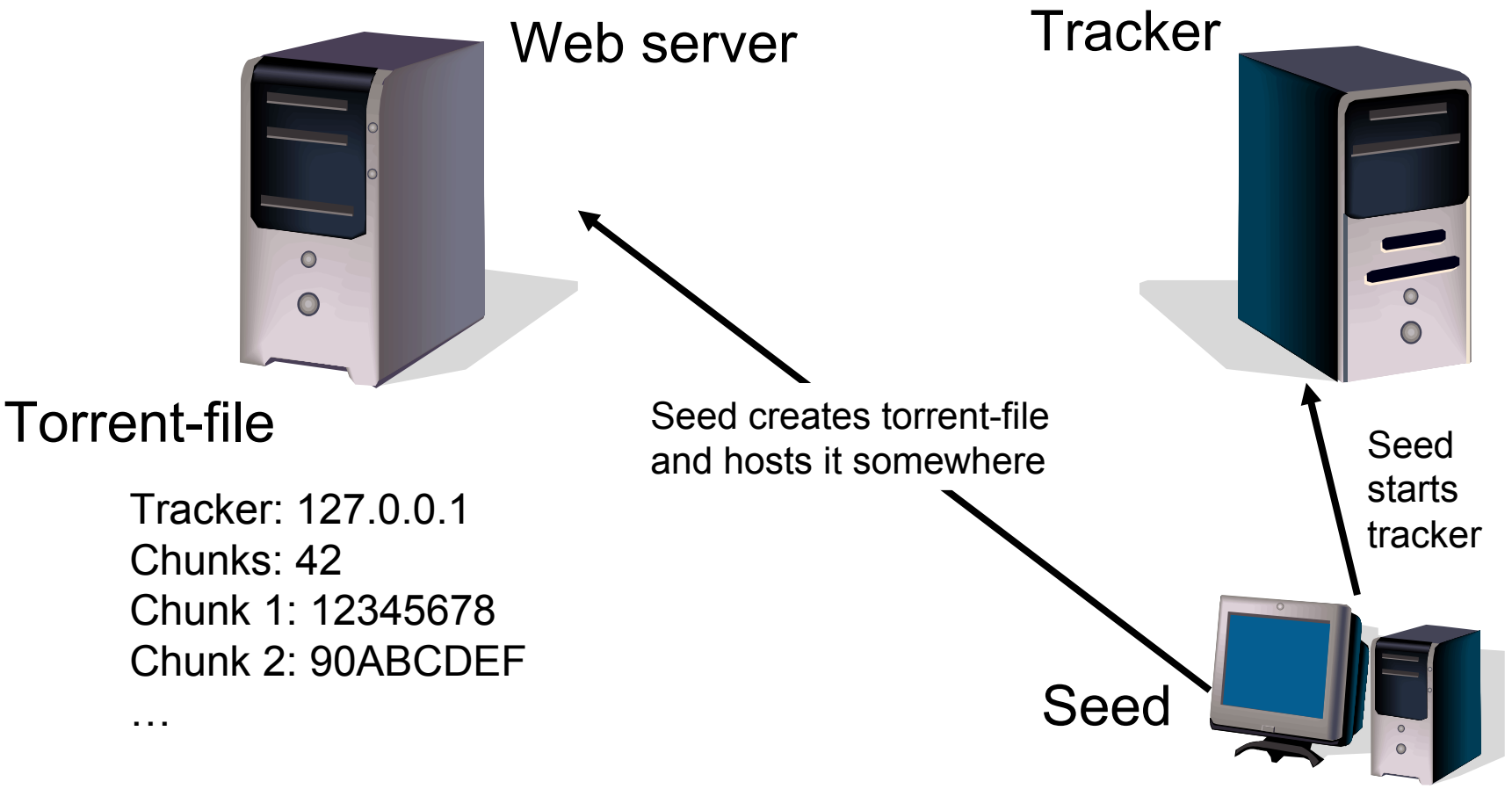
- BitTorrent developed by Bram Cohen in 2001
  - Written in Python, available on many platforms
- Uses old upload/download-ratio concept from BBSs
  - “The more you give, the more you get”
  - Participation enforced in protocol
  - Other P2P systems have adopted similar ideas
- BitTorrent originally used only seldom for illegal content
  - No search functionality?
  - Original source easily identified?
  - Currently lots of illegal content on BitTorrent too (decreasing)…

BitTorrent: How does it Work?

- For each file shared on BitTorrent, there is (initially) one server which hosts the original copy
  - File is broken into chunks
- A “torrent” file which gives metadata about the file
  - Torrent file hosted typically on a web server
- Client downloads torrent file
  - Metadata indicates the sizes of chunks and their checksums
  - Metadata identifies a tracker
- Tracker is a server which tracks currently active clients
  - Tracker does not participate in actual distribution of file
  - Law suits against people running trackers have been successful, even though tracker holds no content (maybe, see Chapter 7)

BitTorrent: Players

- 3 entities needed to start distribution of a file



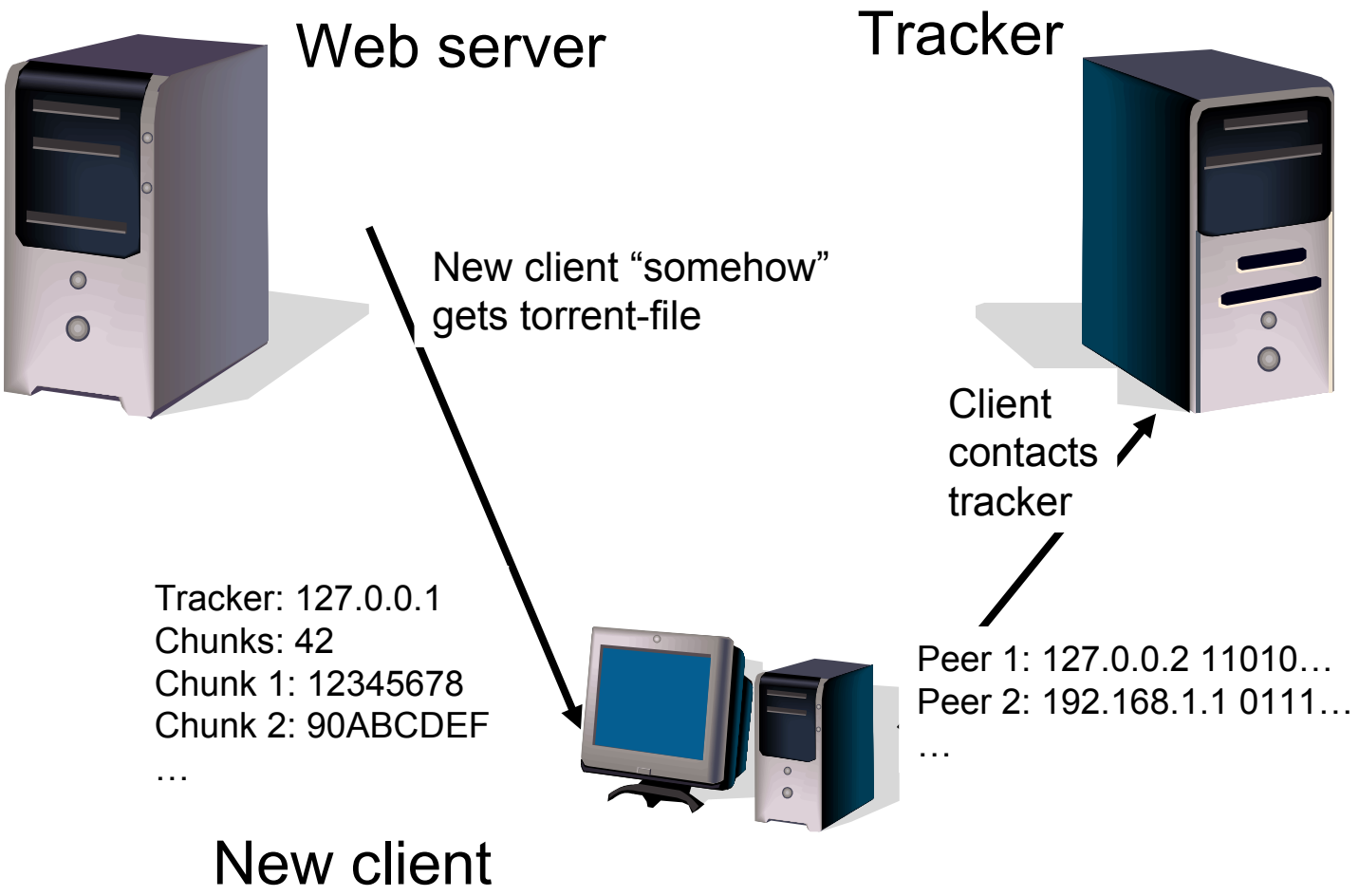
BitTorrent: How does it Work (2)?

- Terminology:
  - Seed: Client with a complete copy of the file
  - Leecher: Client still downloading the file

- Client contacts tracker and gets a list of other clients
  - Gets list of 50 peers and their chunk availability
- Client maintains connections to 20-40 peers
  - If number of connections drops below 20, it contacts tracker
- This set of peers is called peer set
- Client downloads chunks from peers in peer set and provides them with its own chunks
  - Chunk size typically 256 KB
  - Chunks make it possible to download large file in parallel

BitTorrent: Starting Up

- New client gets torrent-file and gets peer list from tracker



BitTorrent: Tit-for-Tat and Chunk Selection

- BitTorrent uses tit-for-tat policy
- A peer serves peers that serve it
  - Encourages cooperation, discourage free-riding
- Peers use rarest first policy when downloading chunks
  - Having a rare chunk makes peer attractive to others
  - Others want to download it, peer can then download the chunks it wants
  - Goal of chunk selection is to maximize availability of each chunk
- For first chunk, just randomly pick something, so that peer has something to share

BitTorrent: Choke/Unchoke

- Peer serves 4 peers in peer set simultaneously
  - Seeks best (fastest) downloaders if it's a seed
  - Seeks best uploaders if it's a leecher
- Choke is a temporary refusal to upload to a peer
  - Leecher serves 4 best uploaders, chokes all others
  - Every 10 seconds, it evaluates the transfer speed
  - If there is a better peer, choke the worst of the current 4
- Every 30 seconds peer makes an optimistic unchoke
  - Randomly unchoke a peer from peer set
  - Idea: Maybe it offers better service
- Seeds behave exactly the same way, except they look at download speed instead of upload speed

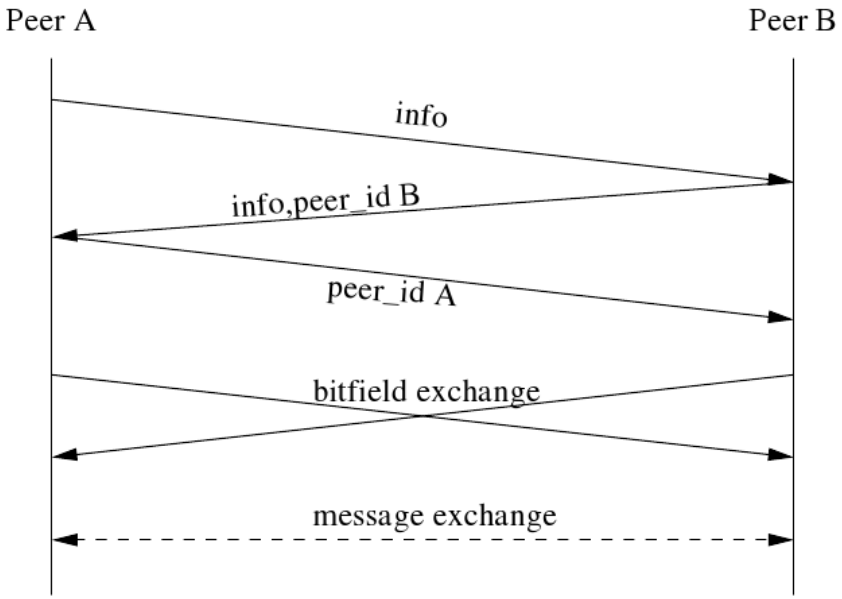
BitTorrent: Choke/Unchoke

- Peer serves 4 peers in peer set simultaneously
  - Seeks best (fastest) downloaders if it's a seed
  - Seeks best uploaders if it's a leecher
- Choke is a temporary refusal to upload to a peer
  - Leecher serves 4 best uploaders, chokes all others
  - Every 10 seconds, it evaluates the transfer speed
- E… 4 …hoke
  - Idea: Maybe it offers better service
- Seeds behave exactly the same way, except they look at download speed instead of upload speed

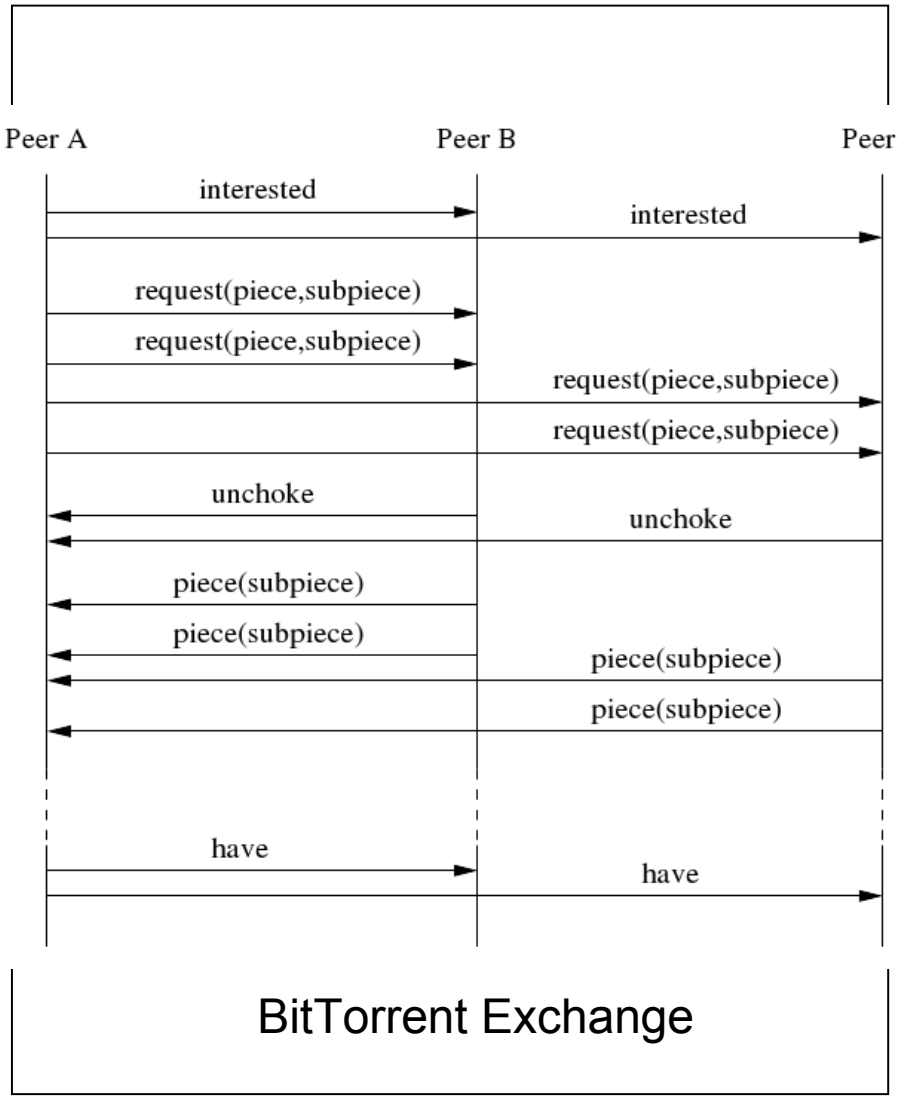
Why only 4?

What happens if one is much slower than all the others?

BitTorrent: a Quick Look at the Protocol

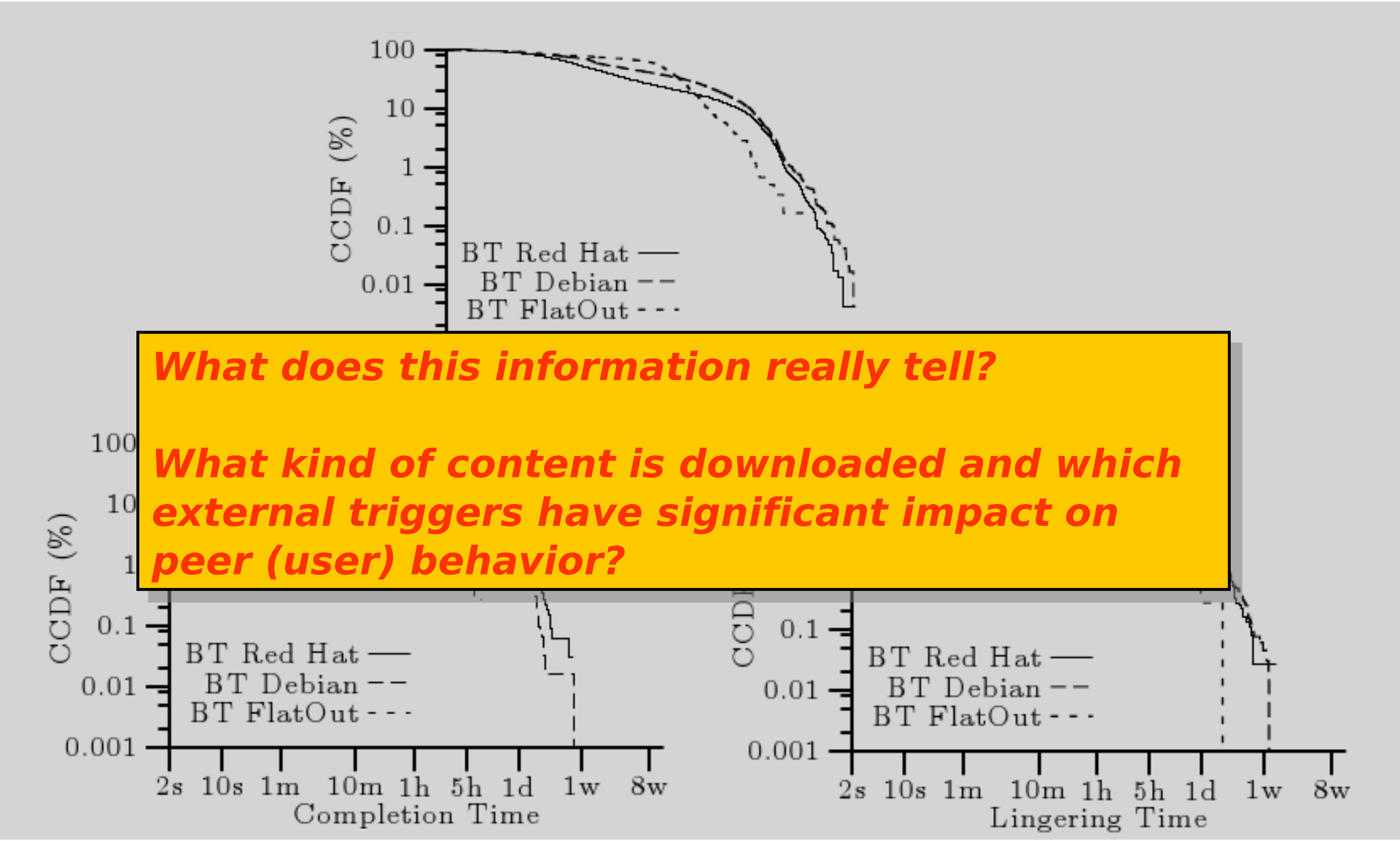


BitTorrent Handshake



BitTorrent Exchange

BitTorrent: How Users Behave



BitTorrent: Strengths

- Works quite well
  - Download a ***bit slow*** in the beginning, but ***speeds up*** considerably as peer gets more and more chunks
- Users keep their peers connected as seeds
  - Legal content, so no need to worry?
  - Large download, leave running over night?
  - How necessary is this?
- Those who want the file, must contribute
  - Attempts to ***minimize free-riding***
- Efficient mechanism for distributing large files to a large number of clients
  - Popular software, updates, …
  - See also Avalanche from Microsoft Research, Pando Media Booster, Akamai's Netsession Interface, BitTorrent DNA

BitTorrent: Weaknesses

- File needs to be quite large
  - 256 KB chunks
  - Rarest first needs large number of chunks
- Everyone must contribute
  - Problem for clients behind a firewall?
  - Low-bandwidth clients have a disadvantage?

BitTorrent: Open Issues

- What is the impact of BitTorrent on the network?
  - Fast download != nearby in network (at least not always)
  - Topic of on-going research
  - First results underline importance of selecting nearby peers for downloading (*)
- What is the optimal chunk selection algorithm?
  - Rarest-first seems to work well in practice
    - Beginning of download, endgame mode, ...
  - Is it also optimal?
  - What is optimal? Fastest for single peer? Overall fastest?
- Is tit-for-tat really necessary?
  - Are there situations where free-riding should be allowed?
  - *Are there situations where free-riding should be encouraged?*

(*)Le Blond, et al.: *Pushing BitTorrent Locality to the Limit*

Freeriders: Problem or Not?

- Freerider is someone who does not contribute
  - Sometimes: Contributes much less than consumes
- Measurement in original Gnutella:
  - 80% of users share little or no files at all
  - Even among the remaining 20%, sharing uneven
- "Rash" conclusion: We must do something about this!
- Sure? Why?
- "Logic": It's not fair!

- True, but is "fairness" the right thing to aim for?
  - How do you define fairness?
- How about optimizing system performance?

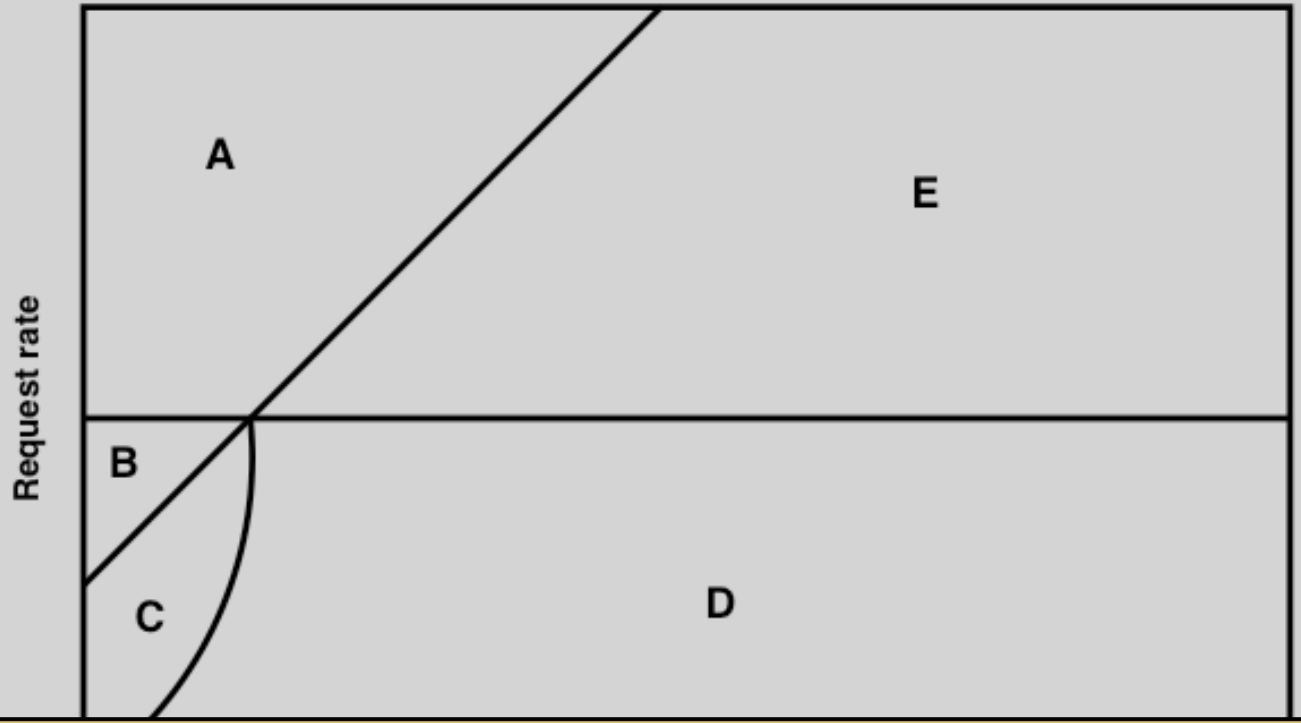
Are Freeriders Really a Problem?

- Short answer: Usually not
- Long answer starts here…
- First, let's look at queueing theory: (classic example)
  - Two printers, fast and slow + standard Poisson assumptions about arrivals and service times
  - ➔You always send print job to fast printer
  - On average you win (as does everyone)
- So what's the relationship to BitTorrent?
- We have two peers: fast and slow
- Where do you want to download from?
- Duh, the fast one of course…
- So: Why should the slow peer even offer the file?

Let's Test This in Practice

- 2 peers, fast and slow, want to download 1 chunk
  - Exponential inter-request times, deterministic service times
  - Model as M/D/1 queue
- Vary arrival and service rates
- Question: How should we split requests between fast and slow peer?
- Can identify 5 possible cases:
  A. Request rate too high to handle, nothing works
  B. Both peers must participate
  C. Every configuration is possible, best if both participate
  D. Every configuration possible, best if only fast sends
  E. Only fast peer is possible

Graphically Speaking



Again: What does this graph really tell?

Could the sizes of the regions differ?
Drastically, even, may be?

- Most of the time we have case D or E (= only fast peer)

Freeriding in General

- Same kind of reasoning can be pushed further
- Three main findings:

1. Freeriding is bad when:
   - Request rate extreme
   - Number of freeriders extreme (over 90%)
2. Freeriding is technically bad, but not noticeable
   - Moderate to high freeriders (50-80%)
   - Increase in download times negligible (~ few % at most)
   - Offered/requested resources homogeneous (only dsl, only dorms)
3. Freeriding is beneficial to everyone
   - Slow (significantly slower!) peers do not offer anything
   - Large gains for everyone!

Freeriding: Recap

- Real-world systems exhibit a lot of free riding
- Gut reaction: Must do something!

- Reality: Not really a major problem to begin with
- Reality: Can even be beneficial

- What happens when fast peers become freeriders?
  - This is of course very bad for everyone…

- Current research: Incentives and cooperation enforcement
- Remember: Forced contributions from everyone not necessary the best thing to do